This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (Previously Presented) A method for accessing data in a database table, comprising:

receiving a fetch request to fetch data from a base table that satisfies a query predicate, wherein rows of the base table are stored in table partitions and wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

determining a set of nodes[[, one]] from [[each] a plurality of index [[partition]] partitions[[,]] whose key column value satisfies the query predicate;

ordering the set of determined nodes from the index partitions;

selecting one node from the ordered set based on a position of the node in the ordering; and

returning data from the table row identified by the location identifier in the selected node in response to the fetch request.

2. (Previously Presented) The method of claim 1, further comprising:

determining whether to modify a direction of the fetch request, wherein the direction comprises the direction in which the index partitions are scanned to determine nodes whose key column values satisfy the query predicate;

modifying the direction of the fetch request if the determination is made to modify the fetch request; and

determining the set of nodes based on the direction of the fetch request.

3. (Original) The method of claim 2, wherein determining whether to modify the direction of the fetch request is based on a current fetch direction and whether the current fetch direction is opposite an ordering of the index partitions.

4. (Previously Presented) The method of claim 2, wherein modifying the direction of the fetch request comprises:

setting the fetch direction to backward if the fetch direction is backward and the fetch direction is not opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is opposite the ordering of the index partitions; and

setting the fetch direction to forward if the fetch direction is backward and the fetch direction is opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is not opposite the ordering of the index partitions.

5. (Original) The method of claim 2, further comprising:

if the fetch request is a first fetch of the fetch request, then selecting one node starting from one of: a lowest key value from each index partition if the fetch direction is forward or highest key value from each index partition if the fetch direction is backward.

6. (Original) The method of claim 2, further comprising:

if the fetch request is not a first fetch of the fetch request, then determining whether a previous direction of a previous fetch request is a same direction as the direction of the fetch request, wherein the direction of the fetch request is capable of having been modified; and

if the previous and current directions are different, then discarding all saved nodes for the index partitions and selecting one node from a last selected node.

7. (Original) The method of claim 6, further comprising:

if the previous and current directions are the same, then scanning in the direction of the fetch request from the previously saved node in each index partition.

8. (Original) The method of claim 1, further comprising:

receiving a subsequent fetch request to fetch data from the base table;

replacing a previously selected node selected in a previous fetch request in the set with one node in the index partition including the previously selected node whose key column value satisfies the query predicate to form a modified set;

selecting one node from the modified set; and

returning the table row identified by the location identifier in the node selected from the modified set.

9. (Original) The method of claim 8, wherein the subsequent fetch request comprises a fetch relative request to fetch a row that is multiple number of rows from the previously selected node, further comprising:

performing the steps of replacing the previously selected node and selecting one node multiple number of times to determine the selected node to return to the fetch relative request to satisfy a fetch quantity.

10. (Original) The method of claim 8, wherein the subsequent fetch request comprises a fetch absolute request to fetch a row that is multiple number of rows from one end of the table, further comprising:

determining a new set of nodes, one from each index partition, by scanning from one end of the index partitions for a first node whose key column value satisfies the query predicate and whose key column value is greater than the previously selected node if fetching forward and the key is less than the previously selected node if fetching backward;

performing the steps of replacing the previously selected node and selecting one node a number of times that is one less than the number of rows indicated in the fetch absolute request to determine the selected node to return to the fetch relative request; and

performing the steps of replacing the previously selected node and selecting one node the multiple number of times to determine the selected node to return to the fetch relative request.

11. (Previously Presented) The method of claim 1, further comprising:

discarding cached keys if the fetch request is in an opposite direction of a previous fetch request;

determining a new set of nodes from each index partition; and

caching the determined new set of nodes when performing the fetch operation.

12. (Previously Presented) The method of claim 11, further comprising:

processing the fetch request to determine set of nodes in the backward direction in the previous fetch request;

inverting the keys and sorting the inverted keys; and

selecting the one node containing the lowest inverted key to return.

13. (Currently Amended) A system for accessing data in a database table, comprising:

a computer readable medium;

a base table implemented in the computer readable medium;

table partitions storing rows of the base table implemented in the computer readable medium;

index partitions, wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

means for receiving a fetch request to fetch data from the base table that satisfies a query predicate;

[[menas]] means for determining a set of nodes[[, one]] from [[each]] a plurality of index [[partition]] partitions[[,]] whose key column value satisfies the query predicate;

means for ordering the set of determined nodes from the index partitions;

means for selecting one node from the ordered set based on a position of the node in the ordering; and

means for returning data from the table row identified by the location identifier in the selected node in response to the fetch request.

14. (Previously Presented) The system of claim 13, further comprising:

means for determining whether to modify a direction of the fetch request, wherein the direction comprises the direction in which the index partitions are scanned to determine nodes whose key column values satisfy the query predicate;

means for modifying the direction of the fetch request if the determination is made to modify the fetch request; and

means for determining the set of nodes based on the direction of the fetch request.

15. (Previously Presented) The system of claim 14, wherein the means for modifying the direction of the fetch request performs:

setting the fetch direction to backward if the fetch direction is backward and the fetch direction is not opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is opposite the ordering of the index partitions; and

setting the fetch direction to forward if the fetch direction is backward and the fetch direction is opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is not opposite the ordering of the index partitions.

16. (Original) The system of claim 13, further comprising:

means for receiving a subsequent fetch request to fetch data from the base table;

means for replacing a previously selected node selected in a previous fetch request in the set with one node in the index partition including the previously selected node whose key column value satisfies the query predicate to form a modified set;

means for selecting one node from the modified set; and

means for returning the table row identified by the location identifier in the node selected from the modified set.

17. (Previously Presented) The system of claim 13, further comprising:

means for discarding cached keys if the fetch request is in an opposite direction of a previous fetch request;

means for determining a new set of nodes from each index partition; and

means for caching the determined new set of nodes when performing the fetch operation.

18. (Previously Presented) The system of claim 17, further comprising:

means for processing the fetch request to determine set of nodes in the backward direction in the previous fetch request;

means for inverting the keys and sorting the inverted keys; and

means for selecting the one node containing the lowest inverted key to return.

19. (Currently Amended) An article of manufacture for accessing data in a database table, wherein the article of manufacture causes operations to be performed, the operations comprising:

receiving a fetch request to fetch data from a base table that satisfies a query predicate, wherein rows of the base table are stored in table partitions and wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

determining a set of nodes[[, one]] from [[each]] a plurality of index [[partition]] partitions[[,]] whose key column value satisfies the query predicate;

ordering the set of determined nodes from the index partitions;

selecting one node from the ordered set based on a position of the node in the ordering; and

returning data from the table row identified by the location identifier in the selected node in response to the fetch request.

20. (Previously Presented) The article of manufacture of claim 19, wherein the operations further comprise:

determining whether to modify a direction of the fetch request, wherein the direction comprises the direction in which the index partitions are scanned to determine nodes whose key column values satisfy the query predicate;

modifying the direction of the fetch request if the determination is made to modify the fetch request; and

determining the set of nodes based on the direction of the fetch request.

21. (Original) The article of manufacture of claim 20, wherein determining whether to modify the direction of the fetch request is based on a current fetch direction and whether the current fetch direction is opposite an ordering of the index partitions.

22. (Previously Presented) The article of manufacture of claim 20, wherein modifying the direction of the fetch request comprises:

setting the fetch direction to backward if the fetch direction is backward and the fetch direction is not opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is opposite the ordering of the index partitions; and

setting the fetch direction to forward if the fetch direction is backward and the fetch direction is opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is not opposite the ordering of the index partitions.

23. (Original) The article of manufacture of claim 20, wherein the operations further comprise:

if the fetch request is a first fetch of the fetch request, then selecting one node starting from one of: a lowest key value from each index partition if the fetch direction is forward or highest key value from each index partition if the fetch direction is backward.

24. (Original) The article of manufacture of claim 20, wherein the operations further comprise:

if the fetch request is not a first fetch of the fetch request, then determining whether a previous direction of a previous fetch request is a same direction as the direction of the fetch request, wherein the direction of the fetch request is capable of having been modified; and

if the previous and current directions are different, then discarding all saved nodes for the index partitions and selecting one node from a last selected node.

25. (Original) The article of manufacture of claim 24, wherein the operations further comprise:

if the previous and current directions are the same, then scanning in the direction of the fetch request from the previously saved node in each index partition.

26. (Original) The article of manufacture of claim 19, wherein the operations further comprise:

receiving a subsequent fetch request to fetch data from the base table;

replacing a previously selected node selected in a previous fetch request in the set with one node in the index partition including the previously selected node whose key column value satisfies the query predicate to form a modified set;

selecting one node from the modified set; and

returning the table row identified by the location identifier in the node selected from the modified set.

27. (Original) The article of manufacture of claim 26, wherein the subsequent fetch request comprises a fetch relative request to fetch a row that is multiple number of rows from the previously selected node, wherein the operations further comprise:

performing the steps of replacing the previously selected node and selecting one node multiple number of times to determine the selected node to return to the fetch relative request to satisfy a fetch quantity.

28. (Original) The article of manufacture of claim 26, wherein the subsequent fetch request comprises a fetch absolute request to fetch a row that is multiple number of rows from one end of the table, wherein the operations further comprise:

determining a new set of nodes, one from each index partition, by scanning from one end of the index partitions for a first node whose key column value satisfies the query predicate and whose key column value is greater than the previously selected node if fetching forward and the key is less than the previously selected node if fetching backward;

performing the steps of replacing the previously selected node and selecting one node a number of times that is one less than the number of rows indicated in the fetch absolute request to determine the selected node to return to the fetch relative request; and

performing the steps of replacing the previously selected node and selecting one node the multiple number of times to determine the selected node to return to the fetch relative request.

29. (Previously Presented) The article of manufacture of claim 19, wherein the operations further comprise:

discarding cached keys if the fetch request is in an opposite direction of a previous fetch request;

determining a new set of nodes from each index partition; and

caching the determined new set of nodes when performing the fetch operation.

30. (Previously Presented) The article of manufacture of claim 29, wherein the operations further comprise:

processing the fetch request to determine set of nodes in the backward direction in the previous fetch request;

inverting the keys and sorting the inverted keys; and

selecting the one node containing the lowest inverted key to return.

31. (New) The method of claim 1, further comprising:

determining whether the key value of the selected node from the ordered set satisfies the query predicate; and

selecting a next node form the ordered set following the selected node that odes not satisfy the query predicate.

32. (New) The method of claim 1, wherein determining the set of nodes from the index partitions comprises executing parallel tasks to process the index partitions.

33. (New) The system of claim 13, further comprising:

determining whether the key value of the selected node from the ordered set satisfies the query predicate; and

selecting a next node form the ordered set following the selected node that odes not satisfy the query predicate.

34. (New) The system of claim 13, wherein determining the set of nodes from the index partitions comprises executing parallel tasks to process the index partitions.

35. (New) The article of manufacture of claim 19, further comprising:

determining whether the key value of the selected node from the ordered set satisfies the query predicate; and

selecting a next node form the ordered set following the selected node that odes not satisfy the query predicate.

36. (New) The article of manufacture of claim 19, wherein determining the set of nodes from the index partitions comprises executing parallel tasks to process the index partitions.